Effective iPhone App Architecture

Doing things right every day

Session 125

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

Effective iPhone App Architecture

Doing things right every day

Matt Drance
Application Frameworks Evangelist

Alexandre Aybes
iPhone Software Engineer

Mac OS X
Mac

2001

3G
9:42 AM
Tuesday
9
Messages
Calendar
Photos
Camera
YouTube
Stocks
Maps
280
Weather
73°
Voice Memos
Notes
Clock
Calculator
Settings
iTunes
App Store
Compass
Phone
Mail
Safari
iPod

2009

```
#warning fix soon

// No idea why this works

#pragma mark unfinished

// Don't hardcode this
```

```
// Pretty sure this leaks

#if 0

// Total mess

#warning magic number
```

Data Persistence

Design Decisions

Memory Management

Application Lifecycle

Data Persistence

Data Persistence

Several kinds of data

- User data
- User preferences
- Sensitive data
- Cached data

Data Objects

Dictionaries gone wild

NSDictionary

```
"sessionID" = "125"
"title"     = "Effective iPhone App Architecture"
"room"      = "Presidio"
"startTime" = "3:30"
...
```

```
label.text = [sessionInfo objectForKey:@"title"];
```

```
[sessionInfo setObject:textField.text forKey:@"title"];
```

```
#define SESSION_TITLE_KEY @"title"
```

Data Objects

Proper data objects

WWDCSessionInfo

```
@property (nonatomic, retain) NSString *sessionID;
@property (nonatomic, retain) NSString *title;
@property (nonatomic, retain) NSString *room;
@property (nonatomic, retain) NSDate *startTime;
...
```

```
WWDCSessionInfo *info = [[WWDCSessionInfo alloc] init];
info.sessionID = @"125";
info.title = @"Effective iPhone Architecture";

sessionDetailViewController.sessionInfo = info;
```

Data Objects

Proper data objects

NSDictionary

```
"sessionID" = "125"
"title"     = "Effective iPhone App Architecture"
"room"      = "Presidio"
"startTime" = "3:30"
...
```

```
label.text = [sessionInfo objectForKey:@"title"];
```

```
[sessionInfo setObject:textField.text forKey:@"title"];
```

```
#define SESSION_TITLE_KEY @"title"
```

Data Objects

Proper data objects

WWDCSessionInfo

```
@property (nonatomic, retain) NSString *sessionID;
@property (nonatomic, retain) NSString *title;
@property (nonatomic, retain) NSString *room;
@property (nonatomic, retain) NSDate *startTime;
...
```

```
WWDCSessionInfo *info = [[WWDCSessionInfo alloc] init];
info.sessionID = @"125";
info.title = @"Effective iPhone Architecture";

sessionDetailViewController.sessionInfo = info;
```

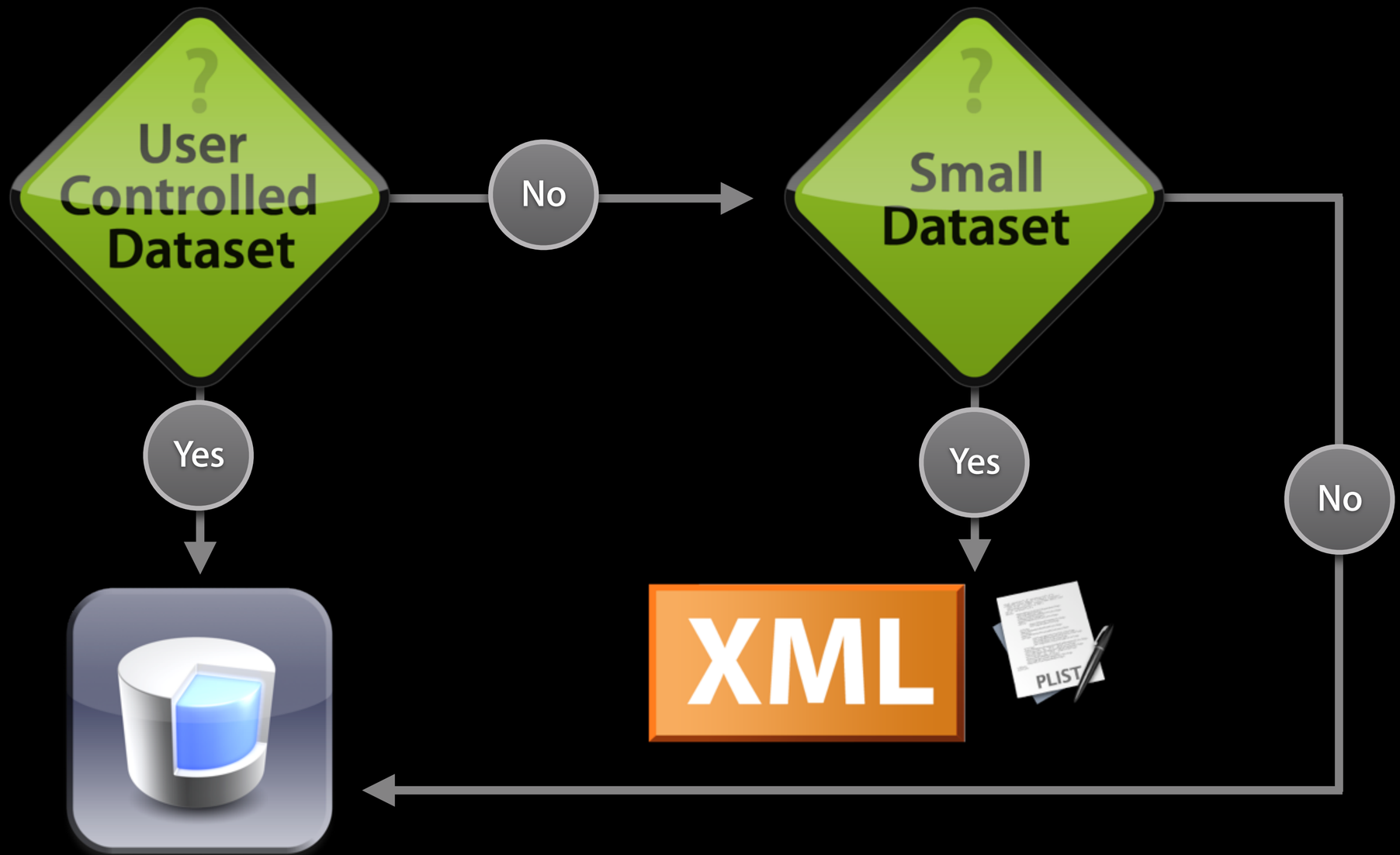
User Data

Multiple options

- Property lists
- XML
- Archiving (NSCoding)
- SQLite
- Core Data

User Data

Choosing your storage



Preferences

NSUserDefaults

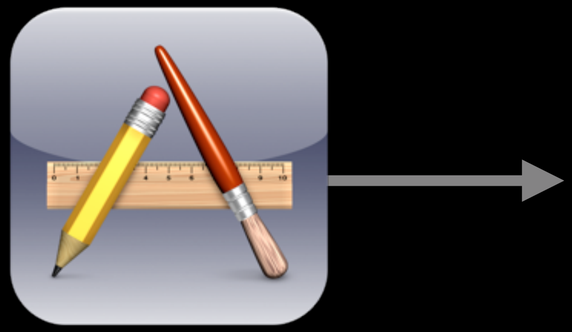
- Lightweight settings
- Navigation state

- Not for heavyweight data
  - No images
  - No massive archived objects

- Not for sensitive data
  - Passwords

Sensitive Data

Keychain



com.paperstreet.Soap

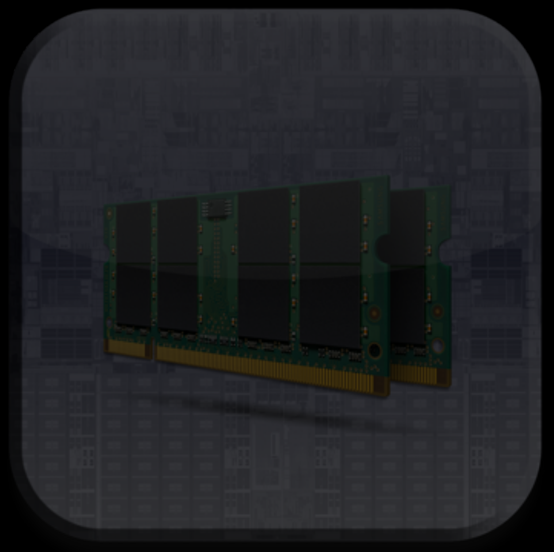
com.paperstreet.Homework

Cached Data

Faster iTunes backups

Data Persistence

Design Decisions



Memory Management

Application Lifecycle

Design Decisions

Alexandre Aybes
iPhone Software Engineer

Massive Changes Made Easier…

Decomposing your app and its workflow

What Are We Going to Cover?

- Application flow
- Information flow

Application Flow

Designing your application

Organizing Content

- Focus on data
- One thing at a time
- Drill down into greater detail

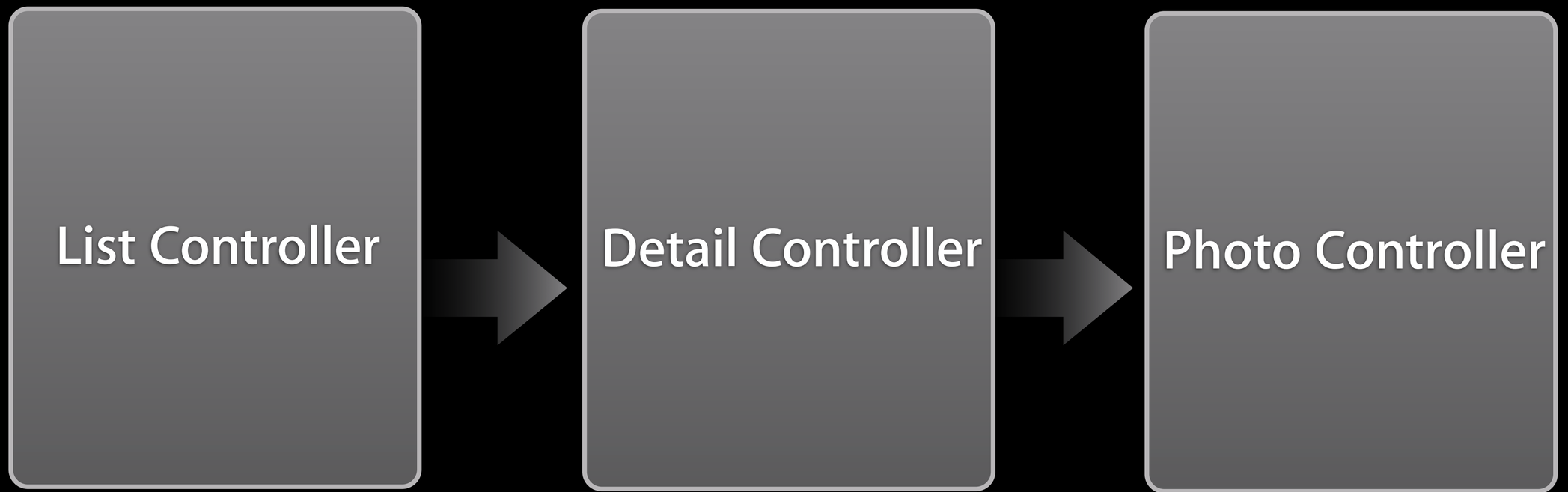
Recipes

Recipe List

Recipe Detail

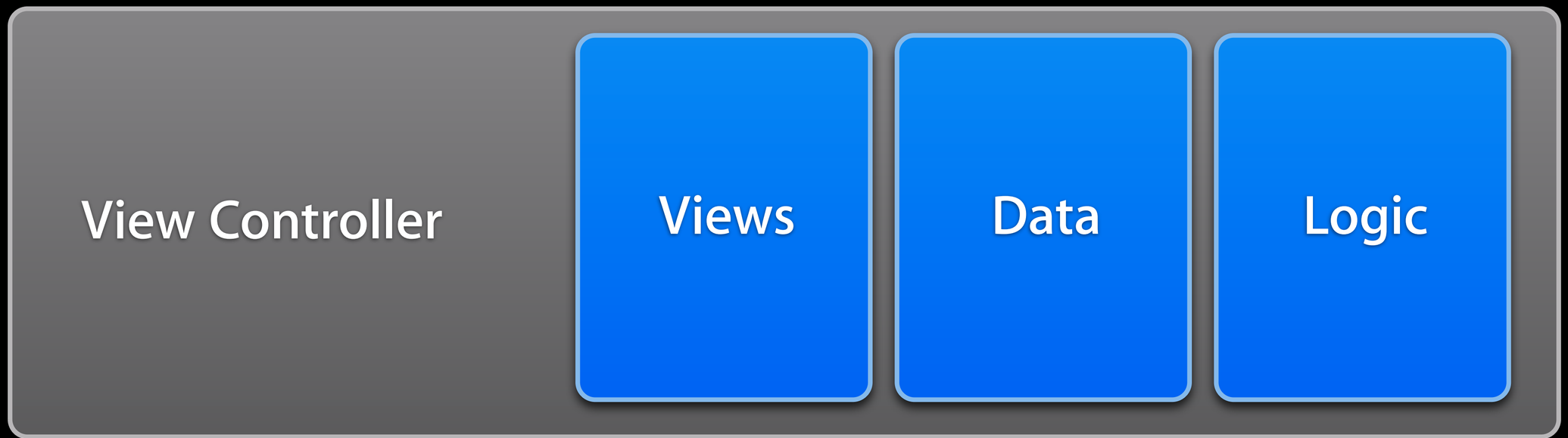
Recipe Photo

A Controller for Each Screen



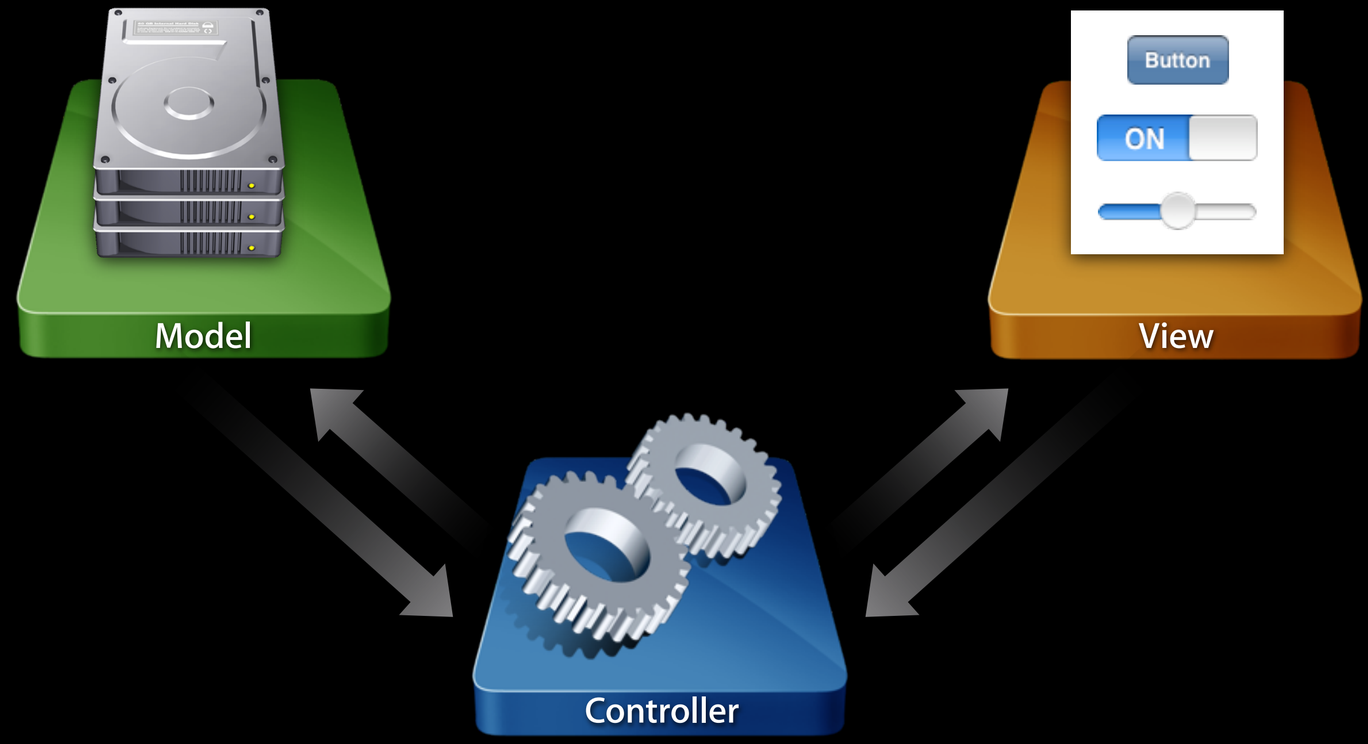
UIViewController

- One screen, one View Controller
- Basic building block
- Subclass to add your application logic



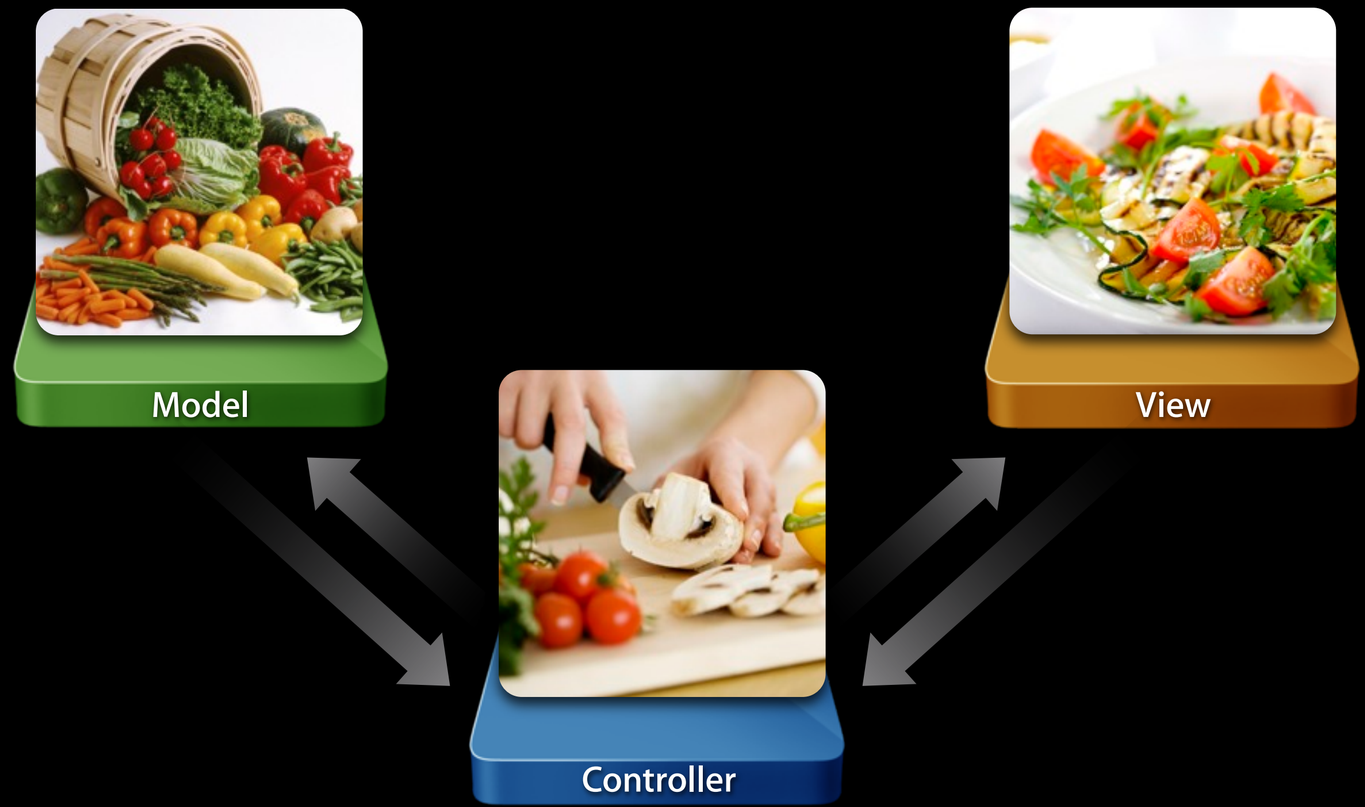
The View Controller's Role

- Load views
- Connect the model to the view
- Manage the flow of the app



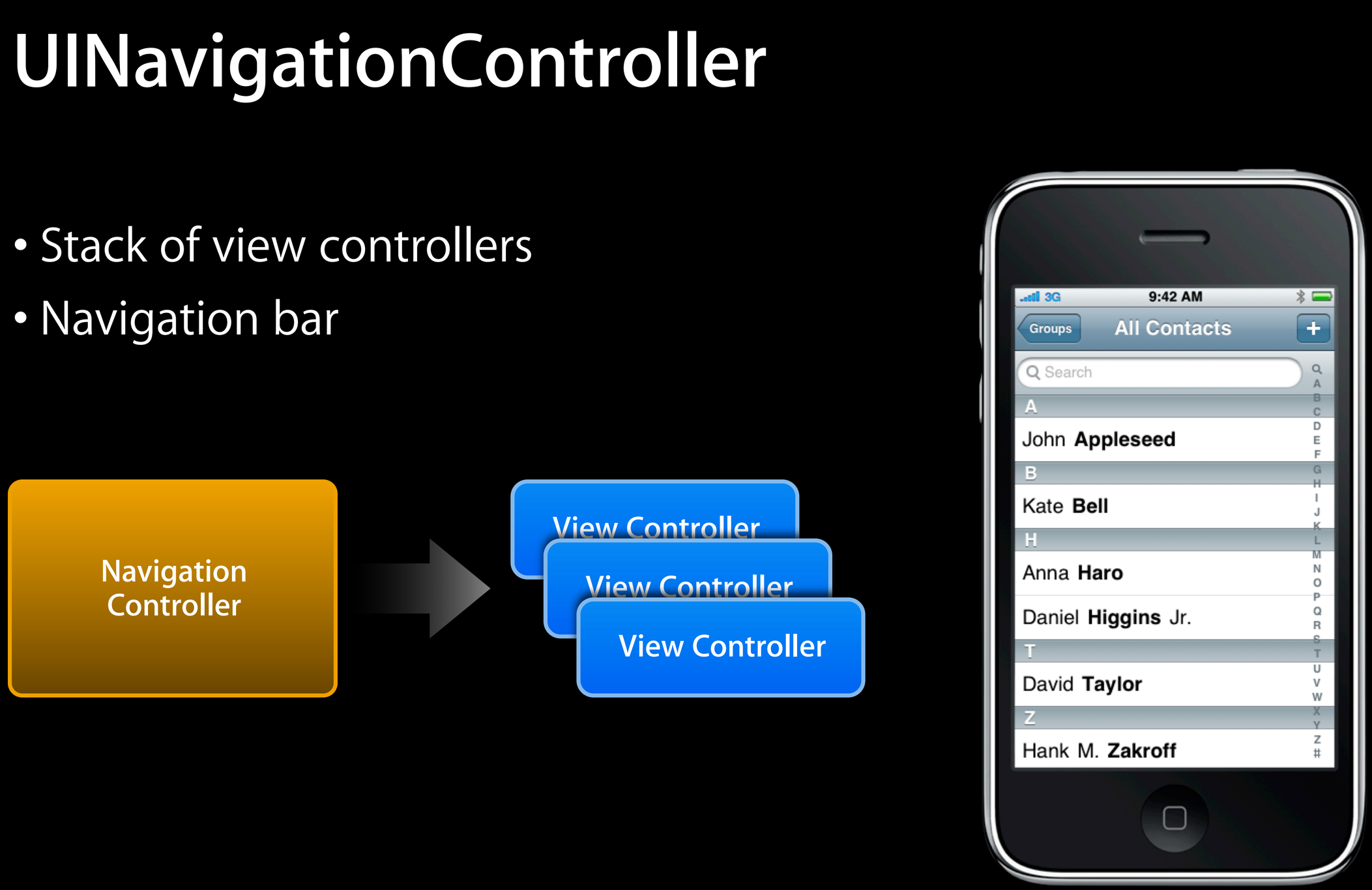
The View Controller's Role

- Load views
- Connect the model to the view
- Manage the flow of the app



UINavigationController

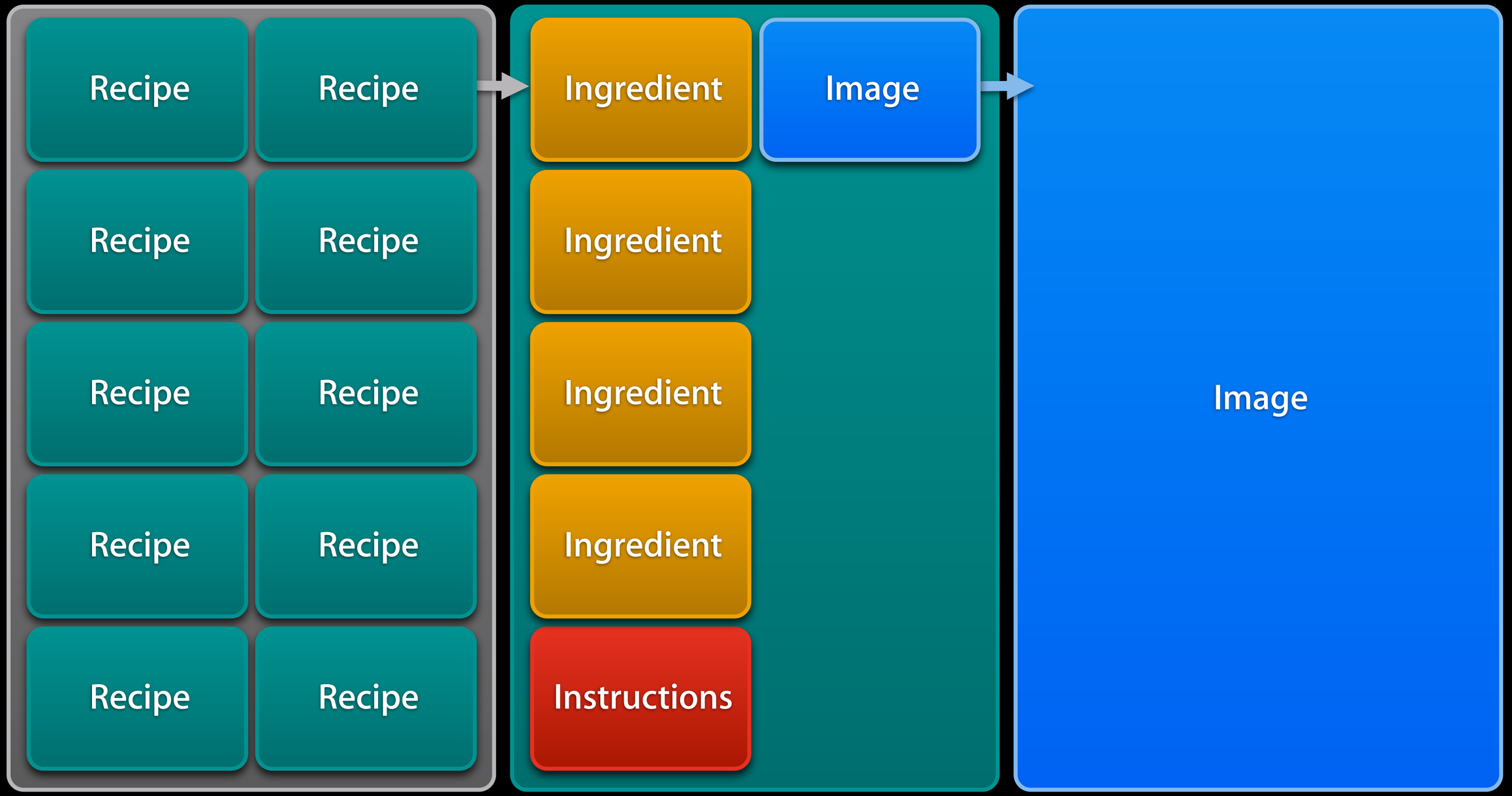
- Stack of view controllers
- Navigation bar



Information Flow

Making it work

The Model


Recipe
Recipe
Recipe
Recipe
Recipe
Recipe
Recipe
Recipe
Recipe
Recipe
Ingredient
Image
Ingredient
Ingredient
Ingredient
Instructions
Image

Many Screens, One Model

The Flow

Delegate

Recipe
3G
9:42 AM
Edit
Recipes
+
Chocolate Cake
1 hour
Chocolate cake with chocolate frosting
Crèpes Suzette
20min
Crèpes flambées with grand marnier.
Gaufres de Liège
1 hour
Belgian-style waffles
Ginger snaps
45 minutes
Nana's secret recipe
Macarons
1 hour
Macarons français: chocolat, pistache, fram…
Tarte aux Fraises
25 min
Delicious tart
Three Berry Cobbler
One and a…
Raspberry, blackbery, and blueberry cobbler
Recipes
Unit Conversion

3G
9:42 AM
Recipes
Tarte aux Fraises
Edit
Tarte aux Fraises
Delicious tart
Preparation time: 25 min
Category
Dessert
Ingredients
One prebaked 9" pastry dough
1
Pastry Cream
1 cup
Strawberries
Recipes
Unit Conversion

3G
9:42 AM
Tarte aux Fraises
Photo

The Application Delegate

- Loads the model
- Hands it off to the first view controller

```
- (void)applicationDidFinishLaunching:(UIApplication *)application {
    vc = [[RecipeListTableViewController alloc] initWithStyle:style … ];
    vc.managedObjectContext = self.managedObjectContext;
    [navigationController pushViewController:vc animated:NO];
 …
}
```

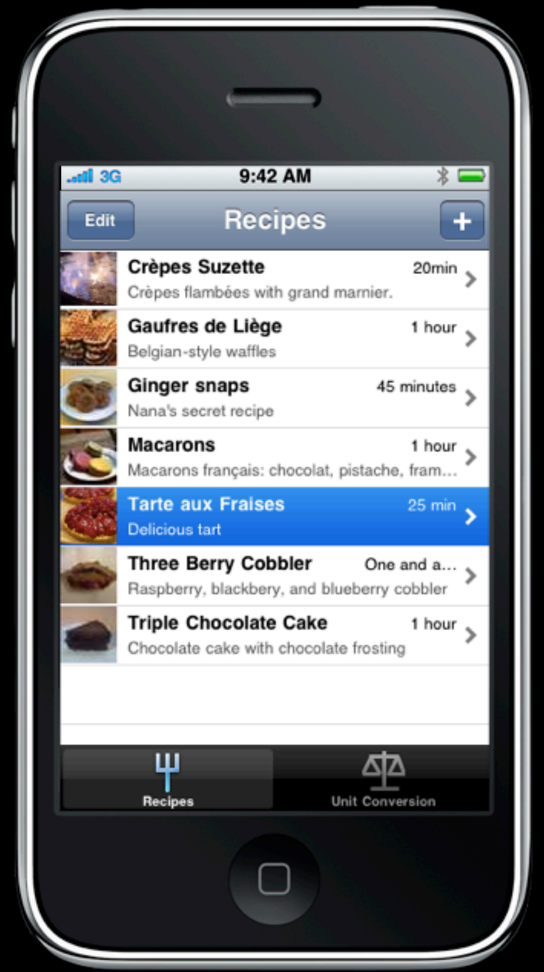
Here Is How It Works

- User Action

```
- (void)tableView:(UITableView *)tv didSelectRowAtIndexPath:(NSIndexPath *)ip {
    Recipe *recipe = [fetchedResultsController objectAtIndexPath:indexPath];
    RecipeViewController *dvc = [[RecipeViewController alloc] init…];
   recipeViewController.recipe = recipe;
    [self.navigationController pushViewController:detailViewController … ];
    [recipeViewController release];
}
```

Here Is How It Works

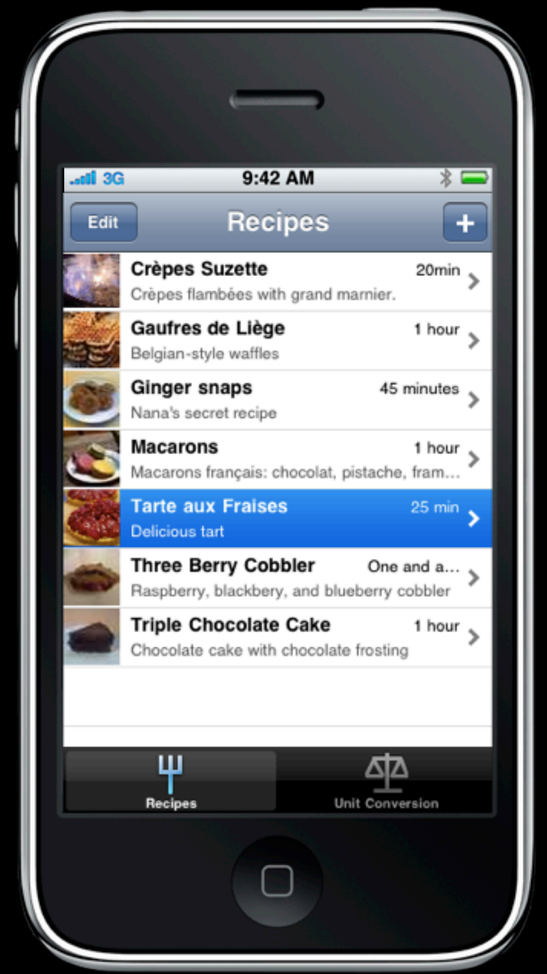
- User Action
- Determine the minimal required data




```
- (void)tableView:(UITableView *)tv didSelectRowAtIndexPath:(NSIndexPath *)ip {
    Recipe *recipe = [fetchedResultsController objectAtIndexPath:indexPath];
    RecipeViewController *dvc = [[RecipeViewController alloc] init…];
    recipeViewController.recipe = recipe;
    [self.navigationController pushViewController:detailViewController … ];
    [recipeViewController release];
}
```

Here Is How It Works

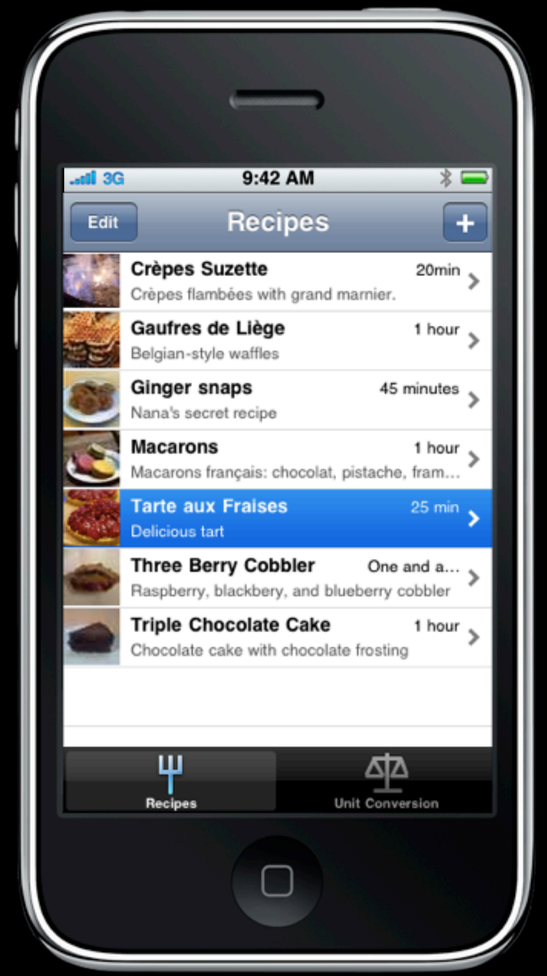
- User Action
- Determine the minimal required data
- Create




```
- (void)tableView:(UITableView *)tv didSelectRowAtIndexPath:(NSIndexPath *)ip {
    Recipe *recipe = [fetchedResultsController objectAtIndexPath:indexPath];
    RecipeViewController *dvc = [[RecipeViewController alloc] init…];
   recipeViewController.recipe = recipe;
    [self.navigationController pushViewController:detailViewController … ];
    [recipeViewController release];
}
```

Here Is How It Works

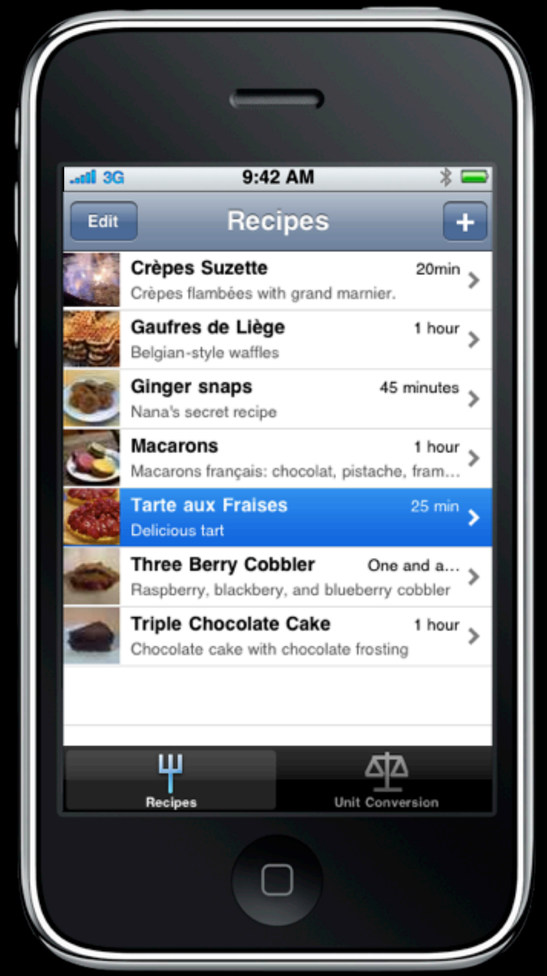
- User Action
- Determine the minimal required data
- Create
- Setup




```
- (void)tableView:(UITableView *)tv didSelectRowAtIndexPath:(NSIndexPath *)ip {
    Recipe *recipe = [fetchedResultsController objectAtIndexPath:indexPath];
    RecipeViewController *dvc = [[RecipeViewController alloc] init…];
   recipeViewController.recipe = recipe;
   [self.navigationController pushViewController:detailViewController … ];
   [recipeViewController release];
}
```

Here Is How It Works

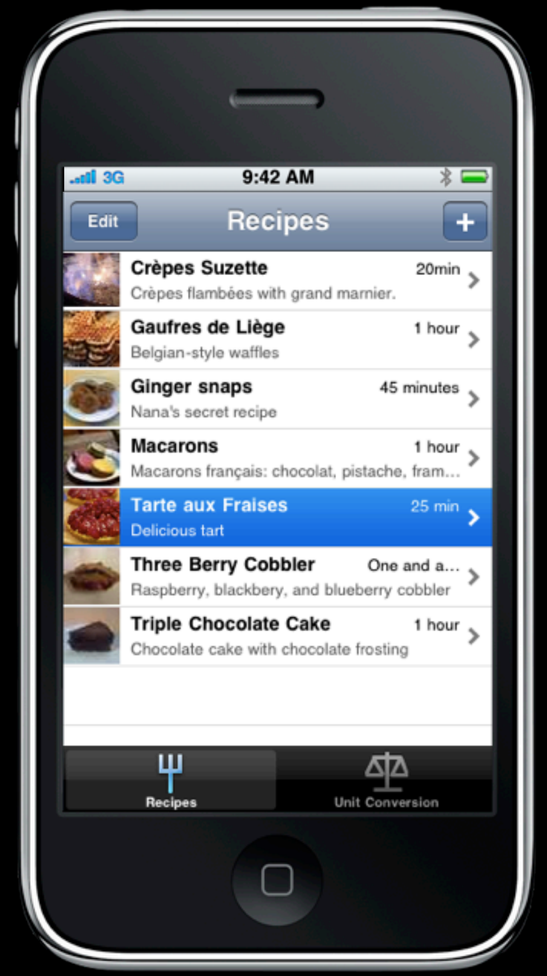
- User Action
- Determine the minimal required data
- Create
- Setup
- Push




```
- (void)tableView:(UITableView *)tv didSelectRowAtIndexPath:(NSIndexPath *)ip {
    Recipe *recipe = [fetchedResultsController objectAtIndexPath:indexPath];
    RecipeViewController *dvc = [[RecipeViewController alloc] init…];
   recipeViewController.recipe = recipe;
    [self.navigationController pushViewController:detailViewController … ];
    [recipeViewController release];
}
```

Here Is How It Works

- User Action
- Determine the minimal required data
- Create
- Setup
- Push
- Forget




```
- (void)tableView:(UITableView *)tv didSelectRowAtIndexPath:(NSIndexPath *)ip {
    Recipe *recipe = [fetchedResultsController objectAtIndexPath:indexPath];
    RecipeViewController *dvc = [[RecipeViewController alloc] init…];
   recipeViewController.recipe = recipe;
    [self.navigationController pushViewController:detailViewController … ];
    [recipeViewController release];
}
```

Forget…Until Something Happens

- Editing
- The parent needs to know
- Delegation

Delegating

- Declare a protocol

```
@protocol RecipeDetailViewControllerDelegate <NSObject>

- (void)recipeDetailViewController:(RecipeDetailViewController*)rdvc
didChangeRecipe:(Recipe*)recipe;

@end
```

- Create a delegate property

```
@property (assign) id <RecipeDetailViewControllerDelegate> recipeDetailDelegate;
```

Being a Delegate

- Implement the protocol

```
@interface RecipeListTableViewController : UITableViewController
<RecipeDetailViewControllerDelegate> {
```

```
- (void)recipeDetailViewController:(RecipeDetailViewController*)rdvc
didChangeRecipe:(Recipe*)recipe {
    [self reloadData];
}
```

- Set the delegate

```
detailViewController = [[RecipeDetailViewController alloc] initWith…];
detailViewController.recipe = recipe;
detailViewController.recipeDetailDelegate = self;
```

Responsible Delegate

- When the delegating object outlives you

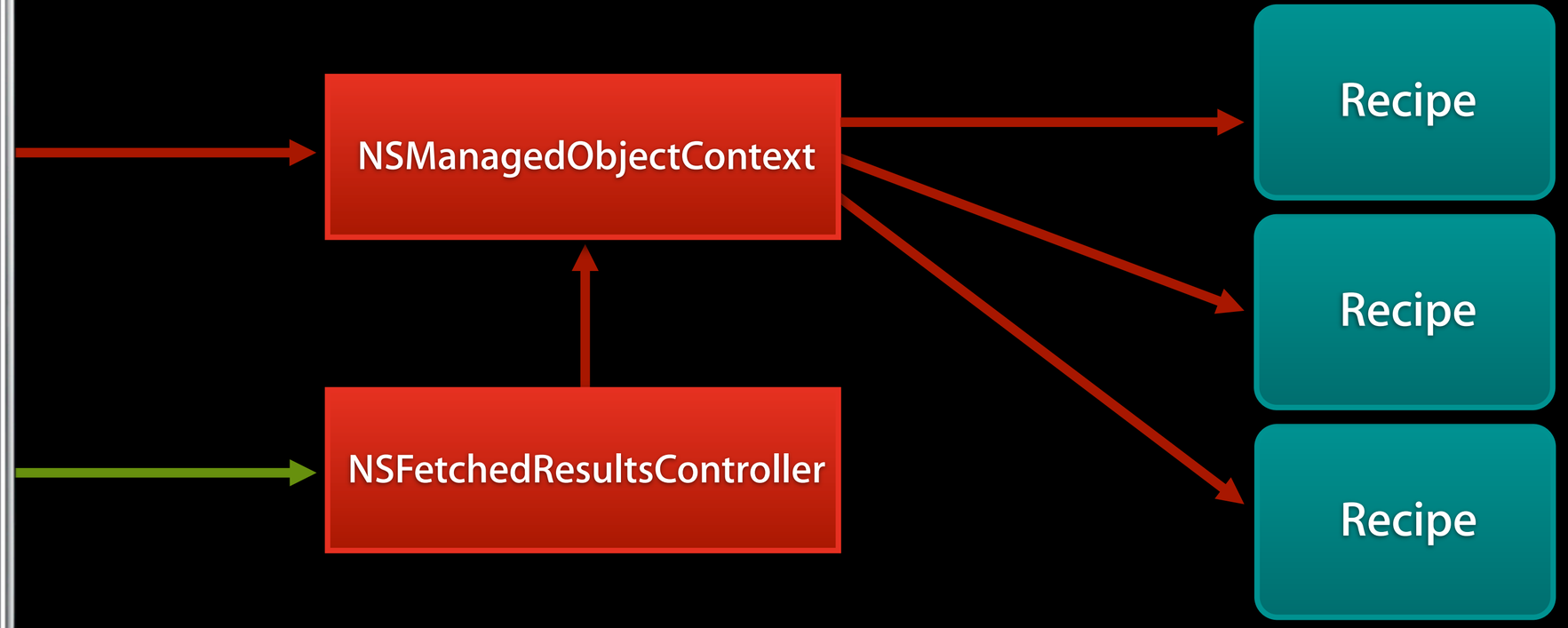
```
- (void)dealloc {
    detailViewController.recipeDetailDelegate = nil;
    // other release
    [super dealloc];
}
```

Alternatives to Delegation

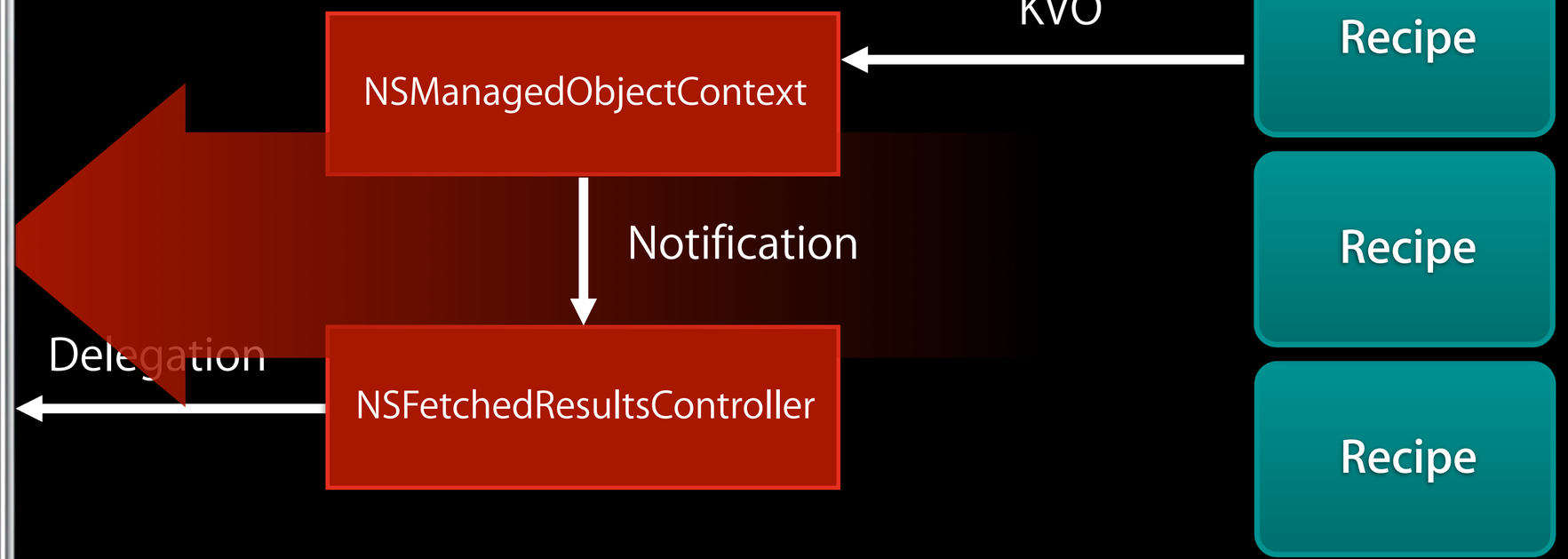
- Notifications
- Key-Value Observing (KVO)

```
- (void)dealloc {
    [[NSNotificationCenter defaultCenter] removeObserver:self];
  // other release
   [super dealloc];
```

Recipes and CoreData



Recipes and CoreData

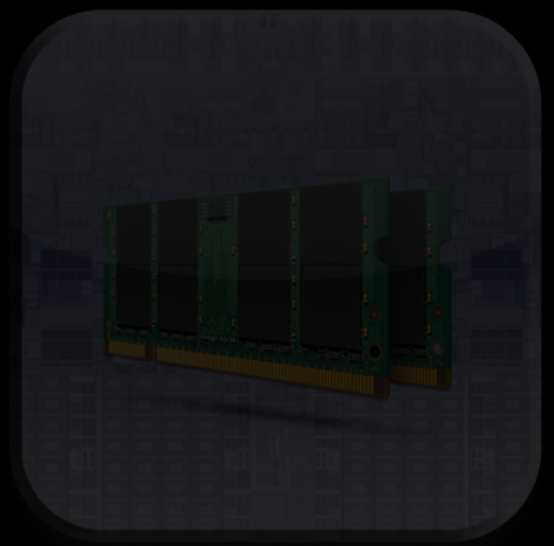


Separation, so What?

- Reuse
- Reorganize

Data Persistence

Design Decisions



Memory Management

Application Lifecycle

Memory Management

It ain't sexy, but it's worth it!

Memory Problems?

- Identifying the problem
- Analysis tools
- Model, views, controllers

Identifying the Problem

The first step towards recovery

Ways to Identify the Problem

- Intermittent crashes
- Crashlogs without backtrace
- Messages in the console

Tools

No, not you.

Instruments

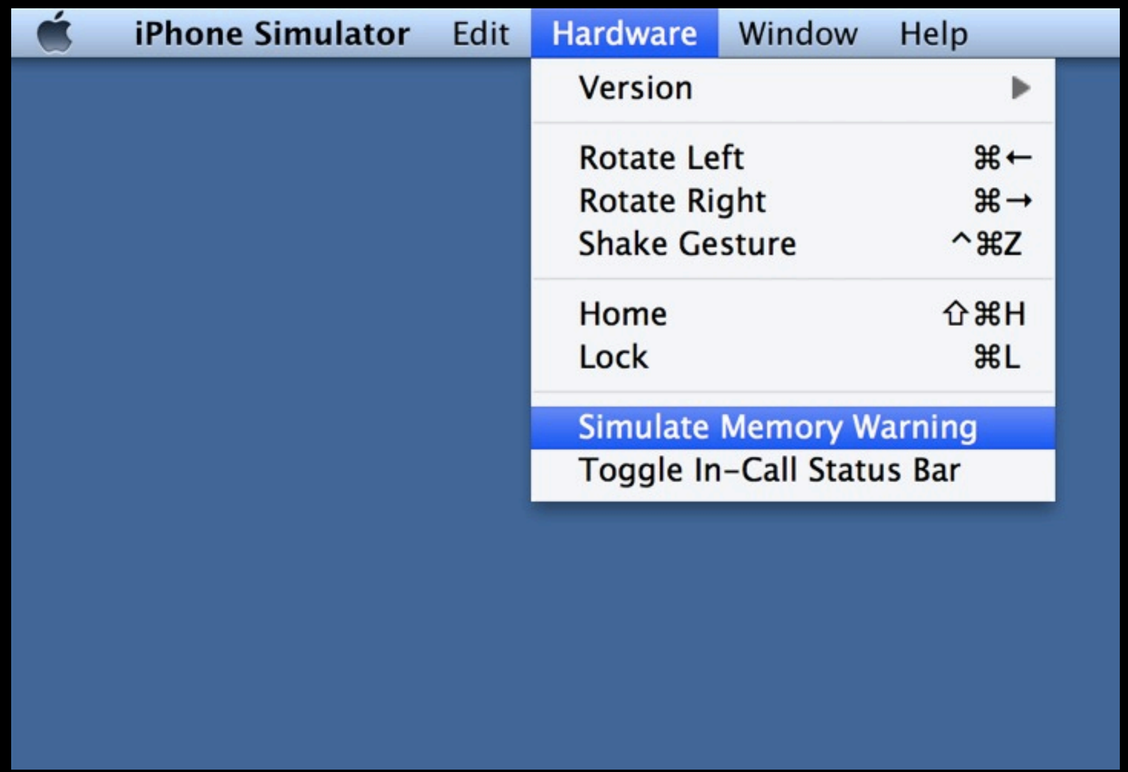
- Leaks

- ObjectAlloc

Understanding Memory Management on iPhone	Mission Thursday 03:30PM
iPhone Performance Optimization with Instruments	Marina Thursday 5:00PM

Simulator

- Memory warnings
- Instruments with the simulator



Xcode Static Analysis

- In Snow Leopard
- Retain/release problems
- Trust, but verify

Demo

Model, Views, Controllers

How can they help?

View Controllers

- No need to cache them
- Respond to memory warnings

```
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // release unnecessary objects
}
```

```
- (void)viewDidUnload
{
    [super viewDidUnload];
    [myView release];
    myView = nil;
}
```

Model

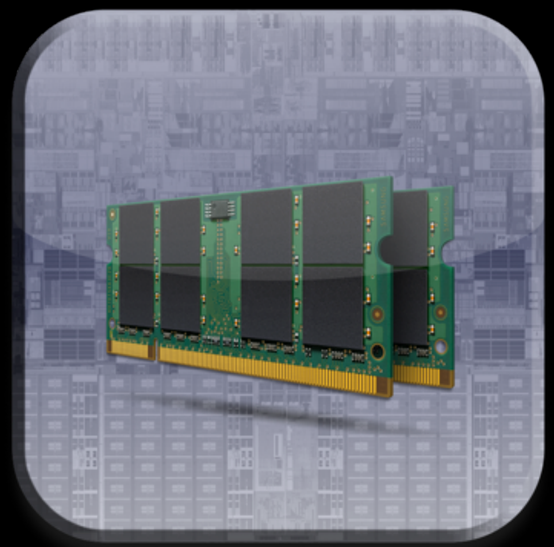
- Spread it thin
- Core Data helps
  - Partial object graph
  - Automatic faulting
  - Responds to memory warnings

Memory Management

- Memory is important
- The tools are here to help
- Architecture makes a difference

Data Persistence

Design Decisions



Memory Management

Application Lifecycle

Application Lifecycle

Application Lifecycle

- Interruptions
- Concurrency
- Compatibility

Interruptions

Handling Interruptions

Preserving the user experience

Handling Interruptions

Three delegate messages

- Interruption occurs
  - `applicationWillResignActive:`
  - Pause and save
- User dismisses
  - `applicationDidResumeActive:`
- User accepts
  - `applicationWillTerminate:`
  - Same as pressing home button

Concurrency

Concurrency

Performing side tasks

- Use asynchronous API where available
  - `NSOperation`
  - `NSURLConnection`
  - Much more
- Illustrate activity to the user
  - `UIApplication`
    - `networkActivityIndicatorVisible`
  - `UIActivityIndicatorView`
  - `UIProgressView`

Threads and UIKit

- UIKit only operates on the main thread
  - Including delegate methods
- Use `performSelectorOnMainThread:` when needed
  - Ask NSThread when unsure

```
if ([NSThread isMainThread]) {
  [self stormTheCastle:fun];
} else {
  [self performSelectorOnMainThread:@selector(stormTheCastle:)
                         withObject:nil
                      waitUntilDone:NO];
}
```

UIKit Performance

Proper codepaths

- View controllers
  - Access the view property
    - Don’t call loadView
- Views
  - Call setNeedsDisplay to refresh
    - Not drawRect:
  - Call setNeedsLayout
    - Not layoutSubviews

Compatibility

Classname Prefixes

Prevent namespace collisions

Classname Prefixes

Prevent namespace collisions

Method Names

Avoid underscores

WWDCTableViewCell

```
- (BOOL) _canHas:(id)cheezburger;
```

UITableViewCell

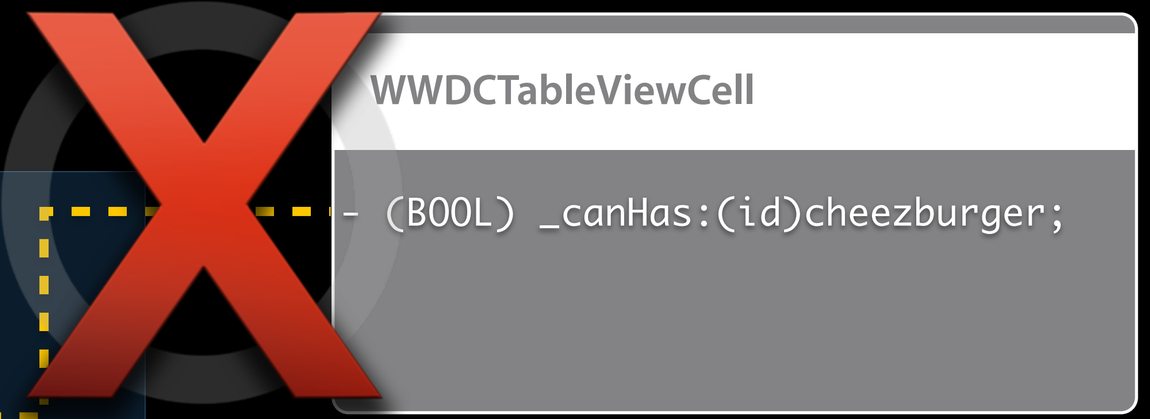
```
- (BOOL) _canHas:(id)cheezburger;
```

Method Names

Avoid underscores

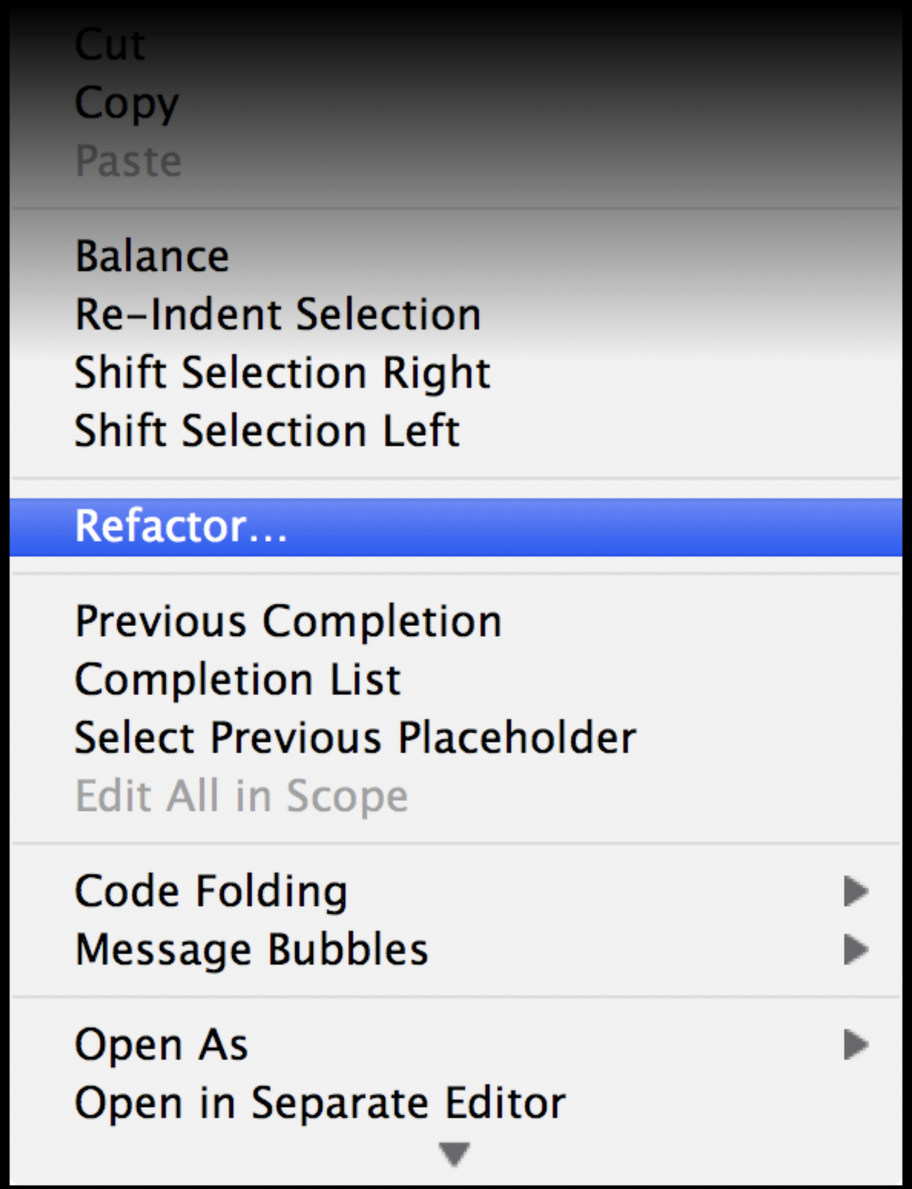
```
// Internal UIKit code

if ([obj _canHas:cbgr]) {
  // ...
} else {
  // ...
}
```



UITableViewCell

```
- (BOOL) _canHas:(id)cheezburger;
```


Cut
Copy
Paste
Balance
Re-Indent Selection
Shift Selection Right
Shift Selection Left
Refactor…
Previous Completion
Completion List
Select Previous Placeholder
Edit All in Scope
Code Folding
Message Bubbles
Open As
Open in Separate Editor

Supporting New Features

Objective-C runtime

```
if ([cell respondsToSelector:@selector(isHighlighted)]) {
    // ...
  } else {
    // ...
}
```

More Information

Matt Drance
Application Frameworks Evangelist
mdrance@apple.com

Documentation
iPhone Application Programming Guide
http://developer.apple.com/iphone

Memory Management Programming Guide for Cocoa
http://developer.apple.com/iphone

iPhone OS 3.0 Readiness Checklist
http://developer.apple.com/iphone

Related Sessions

Understanding Memory Management on iPhone	Mission Thursday 3:30PM
iPhone View Controller Techniques	Presidio Thursday 5:00PM

Summary

- Manage data carefully
- Separate responsibilities
- Conserve memory
- Maintain the experience

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09